A Planning Language for Activity Scheduling

David Zoch, David LaVallee, Stuart Weinstein
Ford Aerospace Corporation
7375 Executive Place, Suite 100
Seabrook, Md 20706

G. Michael Tong
Goddard Space Flight Center
Data Systems Technology Division, Mail Code 522
Greenbelt, Md 20771

Abstract

Mission planning and scheduling of spacecraft operations are becoming more complex at NASA. Spacecraft contain increasingly powerful on board computers which may be commanded to a vast number of modes and configurations. Automated planning and scheduling tools are needed to support the dramatic increase in capabilities, system performance, and user flexibilities. This paper describes a mission planning process; a robust, flexible planning language for spacecraft and payload operations; and a software scheduling system that generates schedules based on the planning language inputs. The mission planning process often involves many people and organizations. Consequently, a planning language is needed to facilitate communication, to provide a standard interface, and to represent flexible requirements. The software scheduling system interprets the planning language and uses the resource, time duration, constraint, and alternative plan flexibilities to resolve scheduling conflicts.

1 Background

NASA performs several types of scheduling. Each type requires different approaches and tools. Types of scheduling include:

- *project scheduling* (e.g., tracking the progress of a project development team)
- *payload manifesting* (e.g., determining the payload manifests for Space Shuttle missions)
- *job shop scheduling* (e.g., refurbishing four Space Shuttles for repeated launches)
- *activity scheduling* (e.g., arranging activities to produce a time line of operations and procedures).

The concepts, approaches, and systems described in this paper apply specifically to activity scheduling, which is a part of the mission planning and scheduling process.

We are concerned with the planning and scheduling of NASA mission operations with respect to spacecraft, flight instruments, space and ground communications networks, and NASA customers (science, application, and commercial users). In our applications, planning consists of deciding what instrument activities, spacecraft activities, and ground activities to perform, while scheduling consists of allocating resources to the activities and sequencing them onto a time line to produce a schedule. Planning is performed by mission planners and science users. Scheduling is currently performed manually with varying degrees of computer assistance but, as we show here, can become highly automated. We focus on the "short term" time frame from four weeks before an activity occurs to the actual real time support of an activity. *Strategic* planning and *tactical* planning involve long term planning conducted months and years in advance and are outside of our planning process except that their products, the mission goals, serve as inputs to our applications.

Activity scheduling includes allocating resources and assigning times to spacecraft and instrument activities. If resources are scarce, one must decide which activities cannot be scheduled. Temporal constraints between activities restrict when activities can be scheduled (e.g., *Activity A* must be scheduled before *Activity B*).

This work was funded by Goddard Space Flight Center under contract NAS 5-31500 with Computer Sciences Corporation

Several techniques are available for generating conflict-free schedules, including hybrid neural network/heuristic approaches as described in [Gaspin, 1989], heuristic approaches as described in [Berner, 1989] and, if the problem is sufficiently constrained, mathematical programming approaches (linear and non-linear) as in [Reddy, 1989]. Techniques for improving an existing schedule include a neural network approach as described in [Sponsler and Johnston, 1990] and a best-first search approach as described in [Odubiyi and Zoch, 1989].

As flexibilities are added to plans, the scheduling procedure becomes more complicated. For instance, if specific resource requirements, start times, and end times for an activity are requested, a scheduling system can respond with a simple yes/no. If flexibilities are specified in the resource requirements and general temporal requirements are specified instead of specific start/end times, the scheduling software must search the current schedule for places where temporal requirements and resource requirements are met. If resource requirements cannot be met, the scheduler can utilize the specified resource flexibilities to determine valid times for scheduling the activity. For example, instead of specifying a request for a 10 minute communication with TDRS-E (Tracking and Data Relay Satellite, East) at 3 p.m. on a certain day, a plan might specify that communication with either TDRS (East or West) is needed between 2 p.m. and 4 p.m.

The Data Systems Technology Division (Code 520) at Goddard Space Flight Center has developed a testbed to investigate the scheduling process for upcoming missions. The testbed includes a mission planning and scheduling system, the Request-Oriented Scheduling Engine (ROSE), that addresses the activity scheduling problem. Spacecraft operation plans are input to ROSE in a robust planning language called the Flexible Envelope Request Notation (FERN).

In this paper we describe (1) the need for increased automation in mission planning and scheduling, (2) the mission planning and scheduling process, (3) the Flexible Envelope Request Notation (FERN), a language for representing user requirements and flexibilities, and (4) the Request-Oriented Scheduling Engine (ROSE), a scheduling system designed to meet the complex planning and scheduling needs of future NASA missions.

2 The Need for Automation

There are several factors motivating the need for increased automation. These factors include the complexity of flight instruments and spacecraft, the need to provide increased flexibility to users, the need for safety, and the support for complex distributed scheduling architectures. Each of these factors is discussed below.

2.1 Complexity of Flight Systems

NASA flight instruments and spacecraft are physically larger and more complex than past space systems. Past space systems were relatively simple since there was no way to repair on board hardware failures. Now, the Space Shuttle crew can repair and service low-earth orbiting spacecraft. Thus, a major obstacle that restrained complexity has been removed for many missions.

Increasingly powerful on board computers have greatly expanded the capabilities of flight systems which may be commanded to a vast number of modes and configurations. Automated scheduling systems on the ground are needed to support the automated flight systems and keep track of operation time lines which contain numerous constraint and activity relationships. Manual scheduling is becoming impractical.

2.2 The Need for Flexibility

Presently, instrument and spacecraft activities are conducted according to an operations time line developed ahead of time. Users want a more flexible approach that allows real time user interactions with instruments. They want the capability to select and perform different activities based on the results of real time telemetry without going through a lengthy re-scheduling process. Scientists often wish to re-plan operations to react to a "target of opportunity" (a rare phenomena such as a large sun flare, volcano, or hurricane). Rapid, safe re-scheduling may be carried out more quickly using automated systems instead of manual methods.

2.3 The Need for Safety

Evaluating the impact of schedule changes is difficult. Automated scheduling systems provide increased flexibility to manage schedule changes while ensuring health and safety of space systems. Automated systems perform constraint checking and produce various reports such as impact evaluation, schedule statistics, and history logs in order to minimize problems introduced by schedule changes.

2.4 Distributed Scheduling Hierarchy

Automated scheduling systems are needed to support remote science users. Instead of depending on a centralized operations control center to operate their instruments, science users may directly control the flight instruments from a university or other home institution. The planning and scheduling capability is no longer centralized in one place; instead, the planning and scheduling capability is distributed between the operations control center and the user sites. Figure 1 and Figure 2 (described below) show examples of distributed planning and scheduling architectures.

The system architecture is hierarchical because the operations control center must schedule space-to-ground communications support with the Network Control Center (NCC). Planning and scheduling systems exist at the Network Control Center, the operations control center, and the customer sites. With such a complex architecture, automated scheduling becomes mandatory.

Figure 1 illustrates the planning and scheduling hierarchy that currently exists. The network level (level 1) contains the Network Control Center (NCC) and the Flight Dynamics Facility (FDF). The NCC is the control center that schedules communication services for spacecraft that use the Tracking and Data Relay Satellite System (TDRSS). The FDF provides orbit, attitude, and navigation products used for generating mission plans and schedules. At the platform level (level 2), spacecraft are controlled and managed by Payload Operations Control Centers (POCCs) or Mission Operations Control Centers (MOCCs). Presently, at the payload level (level 3), the Space Shuttle may contain Spacelab payloads managed by the Spacelab POCC.

The Space Station Freedom environment is an example of a complex, distributed, hierarchical planning and scheduling network. For the Freedom era, the planning and scheduling process will be more automated and distributed in a hierarchy containing additional levels and elements at each level.

Figure 2 illustrates the planning and scheduling hierarchy for the Freedom era. With additional levels and elements, the Freedom era hierarchy is more complex than the current hierarchy. The network level (level 1) is similar to the current configuration. The platform level (level 2) includes the Earth Observing System Operations Center (EOC), the Space Station Control Center (SSCC), and POCCs for various spacecraft. At the payload level (level 3), the Payload Operations Integration Center (POIC) at Marshall Space Flight Center (MSFC) coordinates activities among the international partners [i.e., Japan and the European Space Agency (ESA)] and many Instrument Control Centers (ICCs) for use of the manned base resources. The customer level (level 4) includes the principal investigators and the ICCs for the instruments. The ICCs may support guest investigators and co-investigators who use remote user workstations or instrument support terminals (ISTs) (level 5) to communicate with an ICC.

3 The Mission Planning Process

Figure 3 shows a mission planning process. Once strategic mission goals have been determined at the beginning of a mission, a repetitive planning process occurs, typically on a week to week basis. Investigators and co-investigators generate plans for instrument operations. Specific resource availability profiles may not be known due to security concerns or the commercial proprietary nature of certain payloads.

While investigators are generating instrument plans, spacecraft operations planners are generating plans for maintaining the health and safety of the satellite. These plans include operations such as tape recorder dumps, command loads, and orbit adjustments.

At some point in time, typically a week or two in advance of schedule execution, plans from the investigators are integrated with spacecraft operations plans, and then schedules are produced. Schedules are analyzed by scheduling personnel to verify that mission goals are being met. If the schedule does not adequately meet the mission goals, it can potentially be improved by using the flexibilities specified in the plans (relaxing resource requirements or scheduling alternative activities, for instance). In distributed environments, scheduling personnel may request additional resources from other scheduling sites. After the schedules are produced, they are sent to the investigators. If schedules are not satisfactory, investigators may submit altered plans.

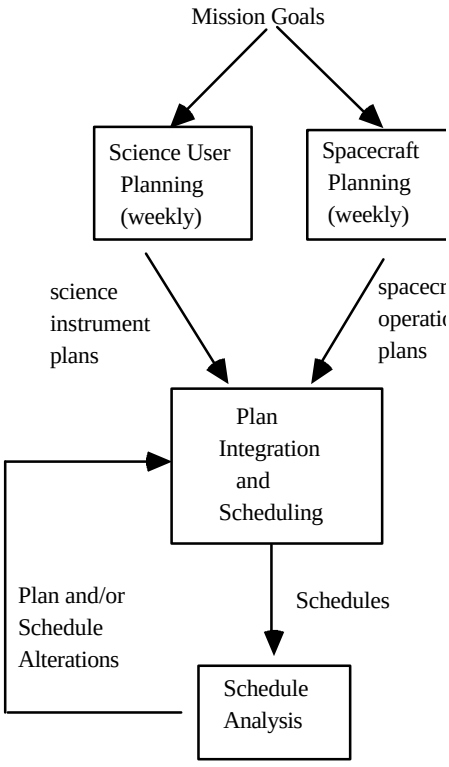

Figure 3. The Mission Planning Process

4 The FERN Language

In an automated scheduling system, users need to express plans for operations in a format that computers can interpret. In general, defining requirements is not simple, whether the requirements describe software functionality, hardware capability, or as in our application, user instrument operations plans and resource needs that support science experiments and flight operations. User resource needs may be complex because user activities are diverse, flexible, and changeable. Their activities may be related to constraints, orbital events, and other activities. A better mechanism is needed to represent this information.

Since people use languages to communicate, we propose that user plans be represented in a language format that computers can process. A language format is needed to express the flexibilities and alternatives contained in the instrument plans. This method is much more expressive than using data structures such as arrays, records, and tables. We use a language format called FERN (Flexible Envelope Request Notation).

FERN has proven to be a general scheduling language. It is has been used to represent Solar Mesosphere Explorer (SME) requests, Upper Atmosphere Research Satellite (UARS) requests, and Network Control Center (NCC) requests.

In many scheduling environments currently in operation, conflicts are often resolved manually. Sometimes users meet to resolve conflicts; however, with increased security restrictions due to DOD and commercial payloads, this form of conflict resolution might no longer be permitted. FERN provides the flexibility that allows an automated scheduling system and project operations personnel to resolve conflicts without violating security restrictions and rules.

FERN supports expressing scheduling requirements at different levels of abstraction. Detailed resource requirements are specified at the lowest level in *steps.* Resource usage within a given step is constant over the duration of the step while the *duration* is often variable. Steps can be grouped together into *activities*. In an operational environment, steps would be defined at the beginning of a mission and then grouped into meaningful activities. Future planning would be done using mnemonic activity names without the need to recalculate detailed step requirements. A pattern of repetition for activities can be specified in a *Generic Request*. A Generic Request can be used to succinctly represent a plan for recurring operations. Each Generic Request is assigned a priority by the user, which indicates its importance relative to other requests by the same user. Temporal constraints can be specified between steps or between activities. Figure 4 shows the organization of information within Generic Requests, Activities, and Steps.

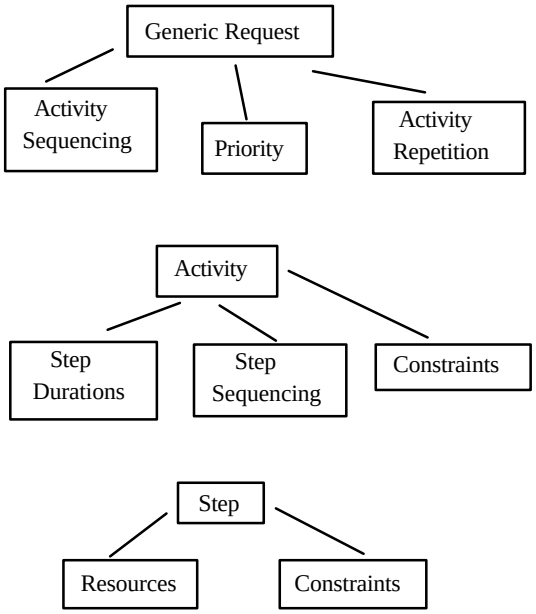

Figure 4. Information Contained in Steps Activities, and Generic Requests

The following sections describe in more detail some of the specific features of FERN. For each requirement in an activity (a resource requirement or a temporal constraint) the user is allowed to specify a *relaxation level* ranking from 1 to 10. If a schedule is generated that is not consistent with mission goals, scheduling personnel will successively relax requirements as specified to attempt to improve the schedule. Requirements with a ranking of "1" will be relaxed first. If no relaxation level is specified, the requirement cannot be relaxed.

4.1 Resource Flexibilities

FERN allows resource amounts to be specified at different relaxation levels. For instance, a power requirement can be specified with two relaxation levels as:

```
POWER  (300 Watts
        AND 250 Watts AT RELAXATION 2
        AND 150 Watts AT RELAXATION 6)
```

In this example, if 300 Watts of power is not available, the scheduling system will try to schedule the request at 250 Watts and then 150 Watts. Requirements with relaxation levels 3, 4, and 5 will be relaxed before the power requirement is relaxed from 250 Watts to 150 Watts.

A non-relaxable requirement is also easy to specify, for instance:

POWER 300 Watts.

4.2 Temporal Expressions

We use the term "interval" to represent a window in time with a specific start and end time and the term "interval set" to represent a collection of non-overlapping intervals. Temporal expressions allow users to create new interval sets as functions of pre-defined interval sets and give names to them such as "weekday" and "spacecraft night." Users may define new interval sets by applying the UNION, INTERSECT, MODIFY, and SELECT operators to existing interval sets.

The UNION and INTERSECT operations are set operators. For instance, given a temporal interval such as "Wednesday" representing a particular 24 hour period and an interval set such as "afternoon" which contains the time period from 1 p.m. to 4 p.m. every afternoon during a week, an interval representing "Wednesday afternoon" could be defined by intersecting "Wednesday" and "afternoon".

The MODIFY operation is useful for changing the start and end times of an existing interval. For example, to create an interval that lasted from 2 p.m. to 3 p.m. using the pre-defined interval above, specify:

```
MODIFY Wednesday-afternoon
            WITH START LATER by 1 hour
            WITH END EARLIER by 1 hour
```

The SELECT operator allows specific windows (intervals) within an interval set to be "selected". For instance, to "select" the second and fourth afternoons from the "afternoon" interval set specify:

```
SELECT afternoon  (2, 4)
```

Temporal expressions are an important tool that enables users to work with their own terminology.

4.3 Temporal Constraints

Once a temporal interval such as "Wednesday-afternoon" is defined it can be used within a temporal constraint. For instance:

```
activity x DURING wednesday-afternoon.
```

FERN contains a general temporal constraint facility for expressing *indefinite interval relations* and the thirteen *simple interval relations* (as described in [Vilian and Kautz, 1986]). Temporal relationships can be specified between two activities or steps. One form a constraint can take is:

```
Request x
  [STARTS | ENDS]
    [MORE THAN | LESS THAN | EXACTLY]
      <duration>
        [BEFORE | AFTER]
          [<activity>| <step>].
```

For instance,

```
Request X starts more than 5 minutes
                before Request Y.
```

Simple interval relations such as "before" and "after" are expressed in a similiar English-like syntax.

4.4 Alternative Activities

Alternative requests allow users to request an entirely different activity if the resource scheduling algorithm cannot accommodate the initial request. Users want to propose alternative experiments if their initial plans cannot be supported.

4.5 An Example Generic Request

To illustrate the hierarchy of the generic request capability a sample set of FERN definitions is shown below. Upper Atmospheric Research Satellite (UARS) contains 10 scientific instruments. One is the Improved Stratospheric and Mesospheric Sounder (ISAMS) which has a 100% duty cycle viewing the Earth's atmosphere limb. There are separate instrument modes for spacecraft day and night. This example only uses some of the expressive capabilities of the language, but it shows the ability of generic requests to dictate many schedule activities over an indefinite period of time:

```
Generic ISAMS_NORMAL_GEN is
   1 ACTIVITY PER UARS_Orbit
   SCHEDULE
      ISAMS_Normal_Act
END GENERIC
```

This example of a generic request definition is straightforward. One occurrence of the activity ISAMS_Normal_Act is to be scheduled every UARS orbit. The activity may turn out to be simple or complex. The activity definition is shown below.

```
ACTIVITY ISAMS_Normal_Act is
   STEP
      ISAMS_Daytime_View_Step
            FOR AS LONG AS POSSIBLE,
      ISAMS_Nighttime_View_Step
            FOR AS LONG AS POSSIBLE
END ACTIVITY
```

This example shows that there are two parts (steps) of the activity. The first step occurs when the spacecraft is in daylight, and the second step occurs during spacecraft

night. The activity definition includes the step durations. A duration of "for as long as possible" needs a constraining time interval. In this case, the constraint is indicated in the step definition:

```
STEP ISAMS_Daytime_View_Step is
  RESOURCES
    ISAMS, -- cold-side limb viewing
    UARS_Power     14 watts
  CONSTRAINT
    Occurs Entirely During UARS_Daytime
END STEP


STEP ISAMS_Nighttime_View_Step is
  RESOURCES
    ISAMS,
            -- cold-side or sun-side limb viewing
    UARS_Power     14 watts
  CONSTRAINT
    Occurs Entirely During UARS_Nighttime
END STEP
```

Steps contain the resource allocations to support the activity. In addition, they may have constraints that restrict when they can be scheduled. The ISAMS_Daytime_View_Step can only occur during the time period defined as UARS_Daytime. ISAMS_Nighttime_View_Step can only occur during UARS_Nighttime. These constraints restrict the actual starting and ending times for the steps. If other requested resources such as UARS_Power are available during the appropriate time periods, the steps are scheduled for the entire duration of the time period UARS_Daytime and/or UARS_Nighttime.

Note that some additional definitions must exist in order to process the above FERN requests. Resource availabilities for ISAMS and UARS_Power must be defined. Time periods for UARS_Orbit, UARS_Daytime, and UARS_Nighttime must also be defined. Since the ISAMS often performs the same science information gathering experiments, these requests can be used repeatedly as needed.

5 The Request-Oriented Scheduling Engine (ROSE)

ROSE is currently under development as a scheduling tool to demonstrate automated scheduling and distributed scheduling concepts. The current major capabilities of ROSE are (1) to receive scheduling messages via a file transfer protocol from any scheduler or user located on the host network and respond with appropriate scheduling messages, (2) to create an initial schedule from user requests, and (3) to re-schedule (as needed) to satisfy mission goals.

ROSE was originally implemented on a Texas Instruments Explorer and has been ported to the Symbolics 36xx environment under Symbolics OS Release 6.1 and Genera 7. The system is currently being ported to Ada in a VMS 5.1 environment using X-windows. We anticipate a port to a UNIX Sun/3 environment so we avoid using any features that would make this port difficult such as VMS system services, implementation-dependent language features, and implementation-dependent X tool kits.

5.1 Communications Capabilities

ROSE supports inter-scheduler communication through the transmission of resource requests and schedules. Users transmit requests expressed in the FERN language to ROSE. The user receives two responses from ROSE. The first is an acknowledgement of the message, confirming receipt of the message. This message indicates whether errors were detected by the FERN parser. When all requests are received, a schedule is created. ROSE sends out schedule messages indicating the name of scheduled requests, the time assigned to the request, and the resource levels dedicated to the request. Users receive a schedule message for each request sent to ROSE, but are not informed of the disposition of requests from other users.

5.2 Scheduling Capabilities

The ROSE system creates an initial schedule from a set of requests, resources, and interactively-specified scheduling heuristics. ROSE currently schedules activities at the rate of approximately 900 activities per hour on a 1 MIP VAX workstation for schedules with 1000 - 2000 requests. The ROSE operator chooses a *selection heuristic* and a *placement heuristic* from pre-defined menus. The selection heuristic evaluates each activity and determines which activity should be scheduled next based on priorities, resource consumption, and an estimation of the restrictiveness of an activity's temporal constraints. The placement heuristic uses activity preferences and information about the existing schedule to determine the placement of the activity. Selection of scheduling heuristics allows creation of many different alternative schedules from the same requests. Alternative schedules can be compared and evaluated with respect to mission goals. ROSE always creates conflict-free schedules. Manual scheduling is also supported through the graphical interface.

5.3 Re-scheduling Capabilities

In a resource constrained environment, resource conflicts will occur, and re-scheduling will usually be a necessary step after the initial schedule is created. A simple approach for scheduling is to resolve resource conflicts by choosing the higher priority activity. In a network of ROSE schedulers, each allowing flexible requests, there are several options:

• *Over book the resource*--in our distributed scheduling environment, over booking is a viable conflict resolution scheme since additional resources can potentially be acquired from another scheduler

• *Relax this activity*--a minor adjustment to the scheduling requirements of the request might allow it to be scheduled

• *Relax other activities*--higher priority activities might have their requirements relaxed in order to accommodate lower priority activities

• *Acquire additional resources*--in a network of schedulers, it might be desirable to request and obtain resources from another scheduler

• *Manually add the activity*-- ROSE provides operator displays and tools that support the interactive re-scheduling of existing activities.

• *Use the automated Schedule Enhancement Technique*-- ROSE provides an automated heuristic search capability similar to a best first search. This technique has proven useful in enhancing existing schedules. The search proceeds by looking for times on the schedule when an activity can *almost* be scheduled, finding those activities that need to be deleted to make it possible to schedule the activity, and then re-scheduling the deleted activities. This technique is described in more detail in [Odubiyi and Zoch, 1989].

• *Choose the higher priority activity*-- as a last resort, some activities are not scheduled.

An operator may re-schedule activities to improve a schedule, to cope with equipment failures, or to accommodate changes in plans. The operator is faced with an overwhelming amount of information. An interactive interface must effectively organize, filter, and display this information at the appropriate level of detail to aid an operator in making informed scheduling decisions. ROSE aids an operator in (1) analyzing and comparing existing schedules, and (2) making modifications to improve a schedule, respond to changes in resource profiles (equipment failures), or respond to new user requests. These features have proven to be a valuable aid in assessing the situation and modifying existing schedules.

5.4 ROSE User Interface

Figure 5 shows the ROSE interface. The three main windows are the Distributed Scheduling Network Window, the Real-Time Message Monitoring Window, and the Timeline of Scheduled Requests Window.

The Distributed Scheduling Network Window displays the scheduling network and the message traffic within the network. Each rectangle represents either a NASA scheduling facility such as a Payload Operations Control Center (POCC) or a user Instrument Control Center (ICC). Figure 5 shows a simplified scheduling network for Space Station Freedom. The Platform Management System (PMS) makes block allocations of resources to scheduling centers P01 and P02. Scheduling requests are sent from the Instrument Control Centers (I01, I02 and I03) to scheduling facilities P01 and P02 where schedules are created. Users at I01, I02 and I03 are then sent scheduling messages that tell them where their requests were scheduled and the amount of resources that were allocated.

The middle portion of the screen is the Real Time Message Monitoring Window. This window displays the names of scheduling messages received by this scheduler from other schedulers in the network. The user can click on these message names to view the details of these messages.

The lower portion of the screen displays the Timeline of Scheduled Requests Window. Schedules are currently one week in duration. Each time line shows the requests scheduled for a particular user instrument. Multiple schedules may be created using different scheduling heuristics. Once created, an operator can rearrange these time lines so that the different schedules can be compared. The operator can also perform standard window manipulations such as panning and zooming on any part of the time line. The Timeline of Scheduled Requests window, in conjunction with the Unscheduled Request Window, provides an object-oriented graphics interface to all requests. Every request can be viewed, edited, relaxed, scheduled, or un-scheduled.

The Timeline of Scheduled Requests Window is also used to display resource plots. Resource profiles can be obtained for original resource amounts, remaining resource

amounts, and resource amounts used by a particular ICC.

During the scheduling process, a ROSE user can set an over booking limit for each resource. This option is useful for investigating the effects of having additional resources. The scheduler may utilize these extra amounts of resources as if they were actually present. As shown in Figure 5, ROSE can display a time line showing where over booked amounts are actually utilized. Clicking the mouse on a rectangle on this time line generates a display showing which resources are over booked at that time.

Figure 6 shows the ROSE interface displaying the results of a "draw available start times" operation. This display gives the operator a complete understanding of why a request could not be scheduled. A time line is displayed for each resource requirement or temporal constraint in the request. The dark areas on the time line show where the particular requirement is satisfied. The top time line, labeled INTERSECTION, displays the intersection of all the other time lines. It shows the places where the request can be scheduled. As shown in Figure 6, the "draw available start times" display contains a time line for the following:

• Every resource or environmental constraint used by the request (power, com-link, tape-recorder, vibration, and NO2-SPEC). The dark areas show places on the time line where sufficient resources are available to meet the needs of this request.

• Each temporal constraints (labeled 1, 2, and 3). The dark areas show places where the temporal constraint is satisfied.

• The DIRECTION constraint (labeled DIRECT). This is a special type of temporal constraint.

Two summary time lines are also shown, labeled DYNAMIC and TEMPORAL. The DYNAMIC time line shows where a request can be scheduled based on its required positioning with respect to other requests. The TEMPORAL time line displays the intersection of all

temporal constraints and the special DIRECTION constraint for the request.

The "draw available start times" display identifies the schedule conflict areas. For example, the N-PROBE-3 request has (1) a small window for the DIRECTION constraint that is restricting the scheduling of this request to a short period early in the schedule, and (2) a conflict in obtaining the NO2-spectrometer instrument resource which is very busy during the early part of this week, making it difficult to schedule the request. Other resources such as "power" and "tape-recorders" are abundant.

6 Conclusion

We have addressed a difficult aspect of mission planning and scheduling--representing the available flexibilities in plans to aid in the automation of the scheduling process and reduce re-planning. The increased automation is necessary to support increasingly complex future NASA missions.

The FERN planning language is designed to be robust, readable, flexible, and object-oriented. FERN supports a variety of user resource requirements and constraints. It supports alternative plans and repetitive activities that are based on temporal expressions (user-defined time periods) rather than specific start times. The language contains hierarchical constructs that support data abstraction and reusable data objects.

References

[Gaspin, 1989] Christine Gaspin. *Mission Scheduling.* Proceedings of the 1989 Goddard Conference on Space Applications of Artificial Intelligence

[Berner, *et al.*, 1989] Carol A. Berner, Ralph Durham, and Norman B. Reilly. *Ground Data Systems Resource Allocation Process.* Proceedings of the 1989 Goddard Conference on Space Applications of Artificial Intelligence

[Reddy, 1989] Surrender Reddy. *Generic Approach to Developing Scheduling Systems*. Computer Sciences Corporation, Beltsville, MD, Document number CSC/TM--89/6106. Prepared for GSFC under contract NAS 5-31500

[Sponsler and Johnston, 1990] Jeffrey L. Sponsler and Mark D. Johnston. An Approach *to Rescheduling Activities Based on Determination of Priority and Disruptivity.* Proceedings of the 1990 Goddard Conference on Space Applications of Artificial Intelligence

[Odubiyi and Zoch, 1989] Jide' Odubiyi and David Zoch. *A Heuristic Approach to Incremental and Reactive Scheduling.* Proceedings of the 1989 Goddard Conference on Space Applications of Artificial Intelligence

[Vilian and Kautz, 1986] Marc Vilian and Henry Kautz. *Propagation Algorithms for Temporal Reasoning*. Proceedings of the Fifth National Conference on Artificial Intelligence